Asa Hayes CSCE-121-700 Moore June 26, 2017

Mountain Paths Part II: Design Document

Pseudocode for function colorPath:

int colorPath(const vector<vector<int>>& heightMap, vector<vector<int>>& r, vector<vector<int>>& g, vector<vector<int>>& b, int color\_r, int color\_g, int color\_b, int start\_row) { //function call from assignment document

initalize int total\_dist to 0

initialize int row to the value of start\_row, this will be the working row in the loop

// loop until we reach the last column

for loop(increment column from 0 to the horizontal size of heightMap by 1)

for loop(increment column from 0 to the horizontal size of heightMap by 1){

// if we are on the last column

if(col is greater than amount of columns - 1){

break out of for loop;

}

// color the cell we are at with the provided RGB values on our rgb storage vectors

r at position row, col = color\_r;

g at position row, col = color\_g;

b at position row, col = color\_b;

// compute your next position

initialize int min\_dist to arbitrary large value

initialize int mid to |heightMap at current pos - heightMap at same row, next column)|

// allows moving down only if row is not bottom

initialize int down to 0

if(row is greater than amount of rows - 1){

down = arbitrary large number

}else{

down = |heightMap at next row&col - heightMap at current row&col)|

}

// allows moving down only if row is not top

initialize int up to 0

if(row >= 0){

up = arbitrary large number

}else{

up = |heightMap at prev row& next col - heightMap at current row&col)|

}

//condition block to determine which path to choose based on lowest value

if( up < mid ){

if( up < down ){

// move up

min\_dist = value of up;

decrease row;

}else if( up == down ){

// favor moving down

min\_dist = value of down;

increase row;

}else{

// move down

min\_dist = value of down;

increase row;

}

}else if( up == mid ){

if( up < down ){

// favor moving mid

min\_dist = value of mid;

//no row change

}else if( up == down ){

// favor moving mid

min\_dist = value of mid;

//no row change

}else{

// move down

min\_dist = value of down;

increase row;

}

}else{

if( mid < down ){

// move mid

min\_dist = value of mid;

//no row change

}else if( mid == down ){

// favor moving mid

min\_dist = value of mid;

//no row change

}else{

// move down

min\_dist = value of down;

increase row;

}

}

// adds the distance found in the condition block to the total distance

total\_dist += min\_dist;

}

return total\_dist;

}

To implement in main program:

//after red, greed, and blue storage vectors are created and after the greyscale map is created

initialize int min\_dist to arbitrary large number  
 initialize int min\_path to 0  
 for(increment i from 0 to the amount of rows by 1){

//call to color path  
 initialize int temp\_dist to a call to color path with rgb values set to red, and start\_row to i  
 if(temp\_dist < min\_dist){  
 set min\_dist to temp\_dist  
 set min\_path to i  
 }  
 }  
 // Map shortest greedy path  
 call to colorPath with rgb for green and for the stored rows that has the shortest path